FAIR: Forwarding Accountability for Internet Reputability

Christos Pappas
ETH Zürich
pappasch@inf.ethz.ch

Raphael M. Reischuk
ETH Zürich
reischuk@inf.ethz.ch

Adrian Perrig
ETH Zürich
adrian.perrig@inf.ethz.ch

***Abstract*—This paper presents FAIR, a forwarding accountability mechanism that incentivizes ISPs to apply stricter security policies to their customers. The Autonomous System (AS) of the receiver specifies a traffic profile that the sender AS must adhere to. Transit ASes on the path mark packets. In case of traffic profile violations, the marked packets are used as a proof of misbehavior.**

FAIR introduces low bandwidth overhead and requires no per-packet and no per-flow state for forwarding. We describe integration with IP and demonstrate a software switch running on commodity hardware that can switch packets at a line rate of 120 Gbps, and can forward 140M minimum-sized packets per second, limited by the hardware I/O subsystem.

Moreover, this paper proposes a "suspicious bit" for packet headers — an application that builds on top of FAIR's proofs of misbehavior and flags packets to warn other entities in the network.

I. Introduction

The frequency and intensity of attacks rooted in misconfigured or vulnerable Internet services has increased in the last months: in February 2014, attackers abused misconfigured time synchronization servers [1] to attack Cloudflare with a peak of 400 Gbps [2]. For 2014, Akamai reports a 90% increase in total DDoS attacks and a 52% increase in average peak bandwidth compared to the previous year [3]. Moreover, man-on-the-side script injection attacks [4] and vulnerable web services have been used as general-purpose attack vectors [5, 6].

These events are explained by the following observations. First, the lack of accountability in today's Internet facilitates attacks with spoofed addresses, allowing attackers to evade blocking mechanisms. Second, the architectural limitations of today's Internet lead to insufficiently effective DDoS defense mechanisms. Third, ISPs have no incentive to punish their misbehaving customers, nor to monitor them. Typically, monitoring comes with high storage and computational requirements that yield additional costs for network operators.

In order to address these problems, the security community has considered several solutions, which come with certain shortcomings: *source accountability schemes* [7, 8] encounter routing scalability problems and introduce prohibitive bandwidth overhead; *cloud-based retroactive DDoS defense services* introduce latency and are insufficiently effective, yet prices can exceed several thousand dollars per month [9]; *capability schemes* [10–13] introduce complexity and require per-flow operations; *extensive filtering* [14–16] requires operator vigilance and out-of-band coordination among ISPs.

Although we stand in solidarity with these proposals, this paper takes a different approach and proposes a lightweight scheme that incentivizes ASes to solve their security problems. To this end, we leverage *forwarding accountability*. In a nutshell, the key idea behind forwarding accountability is to hold ASes accountable for the traffic they forward; transit ASes embed proofs in the packets such that, in case of malicious traffic, a destination AS can later use these proofs to show to the transit ASes that they have indeed forwarded the malicious traffic. We stress that transit ASes do not store any information, but given proofs of misbehavior they can deprioritize traffic from provably malicious ASes. This protects the victim and increases capacity for benign traffic.

We take volumetric DDoS attacks as one possible use case and demonstrate the virtues of forwarding accountability. Consider the topology depicted in Figure 1 and assume web servers, or even servers of critical infrastructures, are located inside AS_n. We assume, exactly as happened in 2014 [2], that an attacker launches a reflection attack against the victims by exploiting the NTP protocol running on misconfigured servers. More precisely, the attacker fakes the victim's source IP address and sends NTP commands to the misconfigured NTP servers within AS_0. The NTP servers reply to the victim with responses that are up to 200 times larger than the initial rogue requests, overpowering the victim's resources. With forwarding accountability in place, the transit ASes embed proofs in the packets that will remind them later that they forwarded the traffic. When the victim reports the attack to transit ASes (AS_1 and AS_2) by providing the proof, the transit ASes acknowledge that they indeed forwarded the malicious traffic. It then becomes evident that AS_0 sourced the malicious traffic, namely from the misconfigured NTP servers. AS_1 can then drop (or at least deprioritize) AS_0's traffic and thus protect not only AS_n and its servers, but also all networks between AS_0 and AS_n. This approach provides benefits also in sparse deployment, where only one transit AS accepts proofs of misbehavior and takes action. Hence, adoption does not require coordination among ISPs.

A cost-effective incremental deployment path is critical to the success of any practical security scheme. ISPs' willingness to adopt security mechanisms is motivated by their reputation and the competitive market environment [17], but constrained by the additional expenses and the lack of economic incentives [18]. In addition, recent Internet regulations [19] intend to actively involve ISPs in stopping the dissemination of malicious traffic, thus making security mechanisms a necessity in the near future. Despite regulatory pressure for adoption of security mechanisms, efficiency and incremental deployment remain important properties that drive adoption.

Contributions. This paper proposes an architectural mechanism, FAIR, to achieve Forwarding Accountability for Internet Reputability. The key concept is that transit ASes embed short cryptographic markings in the packets that will later prove to the ASes that they forwarded these packets. In case of malicious traffic, destination ASes can use these proofs to show to transit ASes that they have indeed forwarded the malicious traffic. After acknowledging the proof of misbehavior, the transit ASes can deprioritize traffic from malicious ASes, increasing network capacity for benign sources.

FAIR is founded on a strong threat model where source, destination, and transit ASes can be compromised or malicious. Moreover, FAIR has the following properties:

- low overhead for processing and bandwidth.
- no per-packet and no per-flow state for forwarding.
- simple key management with one shared key between source and destination ASes.
- deployment compatibility with IP networks.
- complementary applicability to DDoS defense schemes.

We have designed and implemented a software switch performing FAIR packet marking that operates at line rate of up to 120 Gbps; it forwards 140M minimum-sized packets per second on a commodity machine, which is currently limited by the hardware I/O subsystem.

With FAIR in place, we reconsider Bellovin's April Fool proposal of the "evil bit" [20] and propose an extension to our proposal, the "suspicious bit": ASes that forward traffic from misbehaving customers mark this traffic as suspicious, informing other entities in the network. The suspicious bit provides a strong incentive for an AS to watch its traffic and mark malicious traffic itself with the suspicious bit, otherwise its upstream ISP may mark all of the AS's traffic as suspicious, thus, causing collateral damage to benign senders.

II. Overview of FAIR

Before describing our assumptions and protocol details, we first present a high-level overview of FAIR. Our proposal combines ideas from capability systems [10–12] and traceback mechanisms [21], yet its approach is fundamentally different: instead of carrying capabilities, packets collect proofs that will remind transit ASes of having forwarded these packets. In case of malicious traffic, the destination AS sends the proofs *back* to the transit ASes. Communication under FAIR proceeds in three phases. These are depicted in Figure 1 using a line-network topology with cooperating ASes (gray circles) and non-cooperating ASes (black circles)[1]. *Cooperating ASes* are ASes that support FAIR, which, however, does not imply benign behavior.

- **Phase 1 (Setup):** Source and destination ASes set up a communication channel and agree on a sending policy that governs the aggregate traffic from the source AS to the destination AS over a specific AS path. Such a policy can specify the average sending rate, the maximum burst size, or even forbid abnormal packet headers that are used for OS fingerprinting and flooding attacks (e.g., Christmas tree packets [22]).
- **Phase 2 (Transmission):** The source sends data packets to the destination over the communication channel. Each cooperating transit AS inscribes minimal information in the packet headers, which serves as a proof to itself that it has forwarded the packets.
- **Phase 3 (Protest):** If the destination AS detects a policy violation, it proceeds to the protest phase and provides the sending policy together with the data packet headers to the transit ASes, as a proof of misbehavior. This proof of misbehavior identifies the adversary.

[1]This is a simplified communication model, which assumes that all flows from the source to the destination AS follow the same AS path. We will relax this assumption later.

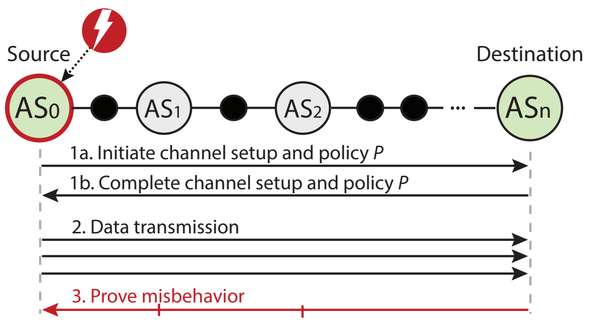
Source
Destination
AS0
AS1
AS2
ASn
1a. Initiate channel setup and policy P
1b. Complete channel setup and policy P
2. Data transmission
3. Prove misbehavior

Fig. 1: Communication under FAIR.

Setting up a sending policy specifies the sending properties of the aggregate traffic from the source AS to the destination AS. A violation implies that the source AS is compromised, malicious, or has poor security practices. A destination AS, depending on its security policies, can drop traffic from source ASes that do not set up a sending policy. Transit ASes receive the proof of misbehavior and can deprioritize inappropriate traffic, depending on their policies. In the DDoS use case, a destination AS establishes a traffic profile with its source ASes and specifies the receiving rates according to its resources. Hence, in case of an attack, the destination AS can prove to transit ASes the sending rate violations.

A. Setup (Phase 1)

Source and destination ASes set up a channel with a sending policy P for traffic from the source AS to the destination AS. The sending policy is formally expressed by the Token Bucket (TB) parameters [23] that the source AS should use for traffic shaping towards the destination AS. In the TB algorithm, a fixed-sized bucket is filled with tokens at a certain rate. A token represents a permission to send a specific number of bits. For a packet transmission, a number of tokens equal to the packet size is removed from the bucket. If there are not enough tokens, the packet either waits for more tokens (shaper) or is discarded (policer). The TB is the formal description of the properties of a transmission. It allows burstiness, but bounds it, as the maximum burst size is proportional to the bucket size.

More specifically, the destination AS specifies two parameters: the Committed Information Rate (CIR), i.e., the average amount of data sent per time unit; and the Committed Burst Size (CBS), i.e., the maximum amount of data that can be sent (for a given time interval). The time interval (T_c) is determined through the relation $CIR = CBS/T_c$. Using these values, the sending policy is then established as follows:

i. The source AS constructs a sending policy packet and sends it to the destination AS.
ii. Each cooperating transit AS indicates its presence on the path. It does not interfere with the sending policy details, nor does it keep per-policy state.
iii. The destination AS completes the sending policy by filling in the CIR and CBS values and returns the information to the source AS.

This is merely an example of a policy construction to demonstrate the necessary information to prove misbehavior in the data plane, which is our focus. For example, to handle temporary increased traffic volumes (e.g., during popular sport events) the source AS can renegotiate the policy's properties and request more bandwidth.

The setup phase can also be substituted by other future Internet proposals. For example, Route Bazaar [24] uses publicly verifiable multilateral contracts among ASes; SCION [25, 26] provides explicit path-validation information for AS paths.

B. Data Transmission (Phase 2)

We describe the data-plane operations performed by source ASes, cooperating ASes, and destination ASes. These operations are applied to each data packet.

Source AS. The source sends data packets over the known path. Border routers of the source AS enforce the sending policy by applying the parameters to the Token Bucket. Moreover, they embed additional information in the packet, including a sequence number and a sending time. This information is used at a later stage to construct a proof of a violation.

Transit ASes. Each egress border router of a cooperating transit AS performs the following operations upon packet reception:

i. The border router verifies that the source's timestamp in the packet is recent and does not deviate from the local time beyond a threshold, otherwise the border router drops the packet.
ii. The border router marks the packet, indicating that it has "seen" the packet. The marking is cryptographically protected with a message authentication code. Since each marking is used to remind only the corresponding AS that inscribed it, it is computed with a secret key that is only known to the AS. This marking is used in the third phase to remind the corresponding AS that it indeed forwarded the packet.

Destination AS. The destination AS monitors the communication channel and performs traffic policing to detect sending policy violations. It stores only packet headers as they contain the markings for the proof of misbehavior, which enables the corresponding transit ASes to acknowledge that they indeed forwarded the packets. If a violation is detected, the destination uses the received packets and proves misbehavior to the transit ASes.

C. Proving Misbehavior (Phase 3)

The goal of Phase 3 is to enable destination ASes to provably protest to other ASes. Taking action against misbehavior is a decision that a destination AS makes according to its interests and policies. Complaints for malicious behavior is an offline procedure between the destination and the transit ASes. The procedure occurs in two rounds.

First, the destination provides the sending policy and the data packet headers to all cooperating ASes on the path. The sending policy contains the transmission properties (CIR and CBS) for the communication channel. The data packet headers contain information for the actual transmission properties. The ASes examine the evidence and acknowledge or reject the complaint. An approved complaint means that the AS acknowledges that it forwarded inappropriate traffic compared to the sending policy specification. This, however, does not mean that the source AS is malicious. For example, if a transit AS injects packets, the source is not responsible for the violation. The destination AS collects approved and rejected complaints from the transit ASes.

In the second round, the destination AS sends all the collected information back to the ASes. Based on this information, the ASes on the path conclude whether the source AS is compromised or whether there were attempts to falsely blame an innocent source. In Section IV-A we explain situations with malicious transit ASes.

III. The FAIR Protocol

We make certain design choices that construct a lightweight accountability mechanism: 1) proofs of misbehavior are carried in data packets, allowing stateless forwarding for transit ASes; 2) probabilistic detection of misbehavior introduces minimal overhead per-packet (a few bytes), keeping bandwidth overhead low; 3) all data-plane cryptography is symmetric, degrading forwarding performance marginally.

A. Assumptions

- *The source knows the AS-level path to the destination and also knows which ASes on the path deploy the mechanism.* BGP update messages contain the AS-level path in the AS-path attribute [27] and cooperating ASes can advertise their support for FAIR in their BGP announcements as a transitive attribute.

- *Participating parties can obtain and authenticate the public keys of all cooperating ASes.* We leverage RPKI [28], a PKI framework that enables entities to authenticate resource certificates (issued by Regional Internet Registries) that bind Autonomous System Numbers (ASNs) to the corresponding public keys, given the correct RPKI public root key.

- *Source and destination ASes perform traffic shaping and policing based on the Token Bucket algorithm.* For

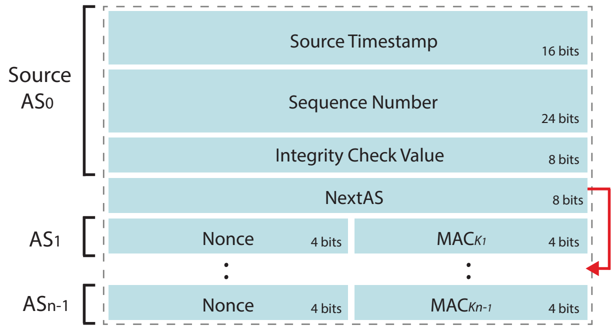

Fig. 2: FAIR Packet Header.

example, Cisco's shaping mechanisms (Generic Traffic Shaping, Class-Based Shaping, Distributed Traffic Shaping) and policing mechanisms (Committed Access Rate, Traffic Policing) are based on the Token Bucket [23].

Furthermore, we assume that the cryptographic mechanisms are secure, i.e., cryptographic hash functions cannot be inverted, signatures cannot be forged, and encryptions cannot be broken.

B. Parameters

Cryptographic Operations. Source and destination ASes establish a secret key (K_{SD}) between them and cache the key to avoid redundant computations. To establish the key, they can obtain the public keys from the RPKI and use a non-interactive Diffie-Hellman key exchange [29, 30]. Furthermore, each transit AS$_i$ uses two local secret keys that can be changed independently from the other ASes: one long-term key for control-plane operations ($\hat{K_i}$) and one key for data-plane operations (K_i). These local secret keys are independent of the communication channels that traverse the AS. Furthermore, transit ASes keep the previous keys for at least $T_m = 12$ hours to be able to verify proof that refers further to the past.

Protest Time Margin (T_m). The destination can protest right after a violation is detected or defer the process to a later point in time. However, we set a time margin after which transit ASes are not obliged to examine proofs of violations, to avoid situations where complaints refer to violations too far in the past. The value for this parameter is agreed upon and universally known to the cooperating ASes. There is no strict requirement for choosing this value; we use $T_m = 12$ hours so that ASes have a loose time window to prove misbehavior.

Clock Deviation. We assume loose clock synchronization between ASes and a reference clock can be set up with an error less than 0.5 seconds; GPS can provide sub-microsecond precision [31]. Furthermore, we assume that the end-to-end packet latency (propagation, transmission, queuing, and processing delay) does not exceed one second.

Reported timestamps in packets are at the granularity of seconds, hence packets with timestamps that differ more than three seconds from the local time at each router are dropped. This check ensures that the timestamps in the packets are fresh and can be used in the protest phase. The three-second margin

Fig. 3: Summary of Symbols and Notation

$P[i]$	Policy packet information inscribed by AS$_i$ on the path.
CIR	Committed Information Rate of the Token Bucket.
CBS	Committed Burst Size of the Token Bucket.
$fair[i]$	FAIR header information inserted by AS$_i$.
PK_i^+/PK_i^-	Public/private key pair of AS$_i$.
K_{SD}	Shared key between source and destination.
K_i	Local secret key of AS$_i$, for data-plane operations.
$\hat{K_i}$	Long-term secret key of AS$_i$, for control-plane operations.
H$(\cdot)$	A collision-resistant hash function, SHA-3.
MAC$_K(\cdot)$	Message Authentication Code using key K.
Sign$_i(\cdot)$	Signature of AS$_i$ with private key PK_i^-.
T_m	Protest Time Margin.
$X\|^{(m)}$	The m Most Significant Bits of X.
$X\|_{(m)}$	The m Least Significant Bits of X.

ensures that packets will not get dropped due to boundary effects when the end-to-end latency and the clock difference add up (one second maximum clock difference between ASes, one second maximum end-to-end latency, and one second due to possible boundary effects during clock transitions).

C. Protocol Operations

We describe the required operations starting with the data plane, which realizes our notion of forwarding accountability. Then, in the control plane, we present a low-latency channel setup and the corresponding sending policy construction. In the end, we show how the sending policy and the data packets are used to prove misbehavior. Figure 3 summarizes the notation we use throughout the paper.

1) Data Plane. First, we show the necessary information and then the interactions between the involved entities. The information and operations described in this section apply to every data packet. Figure 2 shows the corresponding FAIR data packet header.

- Source Timestamp: an indication for the time when the packet has left the source AS. It is a 16-bit value at the granularity of 1 second. It suffices to capture durations up to 18 hours, hence it constrains the possible values for the Protest Time Margin T_m (we have chosen $T_m = 12$ hours).
- Sequence Number: a 24-bit monotonically increasing packet counter inserted by the source AS. The first packet of a communication channel gets the value 0.
- Integrity Check Value (MAC$_{K_{SD}}$): an 8-bit MAC over the payload-length field (in the network-layer header) and the other FAIR related information inserted by the source AS (source timestamp and sequence number). The purpose of the MAC is to signal on-path header modification; it is computed with the shared key K_{SD}. Although the MAC length is short, we do not use the MAC to provide integrity guarantees per packet, but to signal misbehavior over an aggregate of packets. In Section IV-A, we quantify the security implications of this idea. The payload length is included in the computation of the ICV so that the destination stores packet headers only, not the whole packets.

- Nonce fields: a 4-bit value inserted by each AS on the path. It functions as an indicator of having forwarded the packet and to enable detection of replay attacks; the values are chosen uniformly at random.
- MAC fields (MAC$_{K_i}$): a 4-bit MAC inserted by each AS on the path. The input to the MAC is the information that must be integrity-protected to securely prove a sending-rate violation in the protest phase: the packet length in the network-layer header, the source's timestamp and sequence number in the FAIR header, and the nonce field. The local secret key K_i used to compute the MAC is maintained by each AS independently. As described earlier, we use short MACs to signal misbehavior over an aggregate of packets. We will show that even a 1-bit MAC can be used for our purpose (Section IV-A). If a subsequent entity changes any of the previous information in the packet, the MAC verification will fail.
- NextAS: an 8-bit pointer to the position in the FAIR header where the next AS on the path will insert its information. The pointer is initialized by the source and each transit AS modifies it accordingly. The 8-bit field suffices for inter-domain paths up to 256 hops; the average AS-path length today is 3.9 hops (3.5 hops) for IPv4 (IPv6) [32].

The sequence numbers, timestamps, and nonces are used to provide loose replay detection at the AS-level granularity. Replay detection reveals such an attack in the protest phase; the purpose is not to have the destination AS drop replayed packets. The monotonically increasing values of the sequence numbers together with the timestamp values are used to detect replay attacks. Multiple occurrences of a sequence number for the same timestamp reveal the replay. Furthermore, the clock deviation check at each AS hop prevents an attacker from storing and replaying the packet at a later point in time. The random nonces inscribed by each AS provide information about the adversary's position on the path. Nonces localize the adversary to a portion of the path, depending on which nonce fields repeat and which change per replayed packet. Furthermore, the short MAC fields serve as a misbehavior flag (rather than as integrity guarantees per packet): a few verification failures in the protest phase indicate misbehavior. Sections III-D and IV-A provide further details.

Processing of Outbound Packets: The source AS creates a FAIR packet header and fills in its information. The new packet header is placed between the network and transport-layer headers and is created with a sufficient length to accommodate the information of the transit ASes; this ensures that the packet length does not increase en route. The AS-level path is known to the source AS, and each transit AS overwrites 1 byte of the header. Based on the destination address in the packet header, the border router of the source AS determines the shared key with the destination AS, the current packet count for this communication channel ($seqno$), and the output port to forward the packet to. Procedure 1 summarizes the operations that the source performs.

Processing of Forwarding Packets: We describe the actions that each egress border router of the cooperating ASes on the path (AS$_i$, $1 \leq i < n$) performs for each data packet.

Procedure 1: Processing of Outbound Packets

```
procedure SEND($pkt, pkt\_hdr, fair$)
    ▷ $pkt$ refers to the whole packet
    ▷ $pkt\_hdr$ contains the network-layer packet header
    ▷ $fair[\,]$ is the FAIR header that the source creates
    ▷ $cnt$ the packet counter per communication channel
    $(port, K_{SD}, cnt) \leftarrow$ lookup($pkt\_hdr$)
    $fair[0].time \leftarrow$ time()$|_{(16)}$
    $fair[0].seqno \leftarrow$ ++$cnt|_{(24)}$
    $fair[0].icv \leftarrow$ MAC$_{K_{SD}}(pkt\_hdr.payload\_len$
                      $|| fair[0].time || fair[0].seqno)|^{(8)}$
    $fair.nextAS \leftarrow 0$
    transmit($pkt, port$)
```

Procedure 2: Processing of Forwarding Packets

```
procedure FORWARD($pkt, pkt\_hdr, fair$)
    ▷ $pkt$ refers to the whole packet
    ▷ $pkt\_hdr$ contains the network-layer packet header
    ▷ $fair[\,]$ is the FAIR header
    $diff \leftarrow |fair[0].time -$ time()$|_{(16)}|$
    if $diff > 3$ and $diff < 2^{16} - 3$ then
        drop packet
    $fair[i].nonce \leftarrow$ rand()$|^{(4)}$
    ++$fair.nextAS$
    $fair[i].mac \leftarrow$ MAC$_{K_i}(pkt\_hdr.payload\_len || fair[0].time$
                      $|| fair[0].seqno || fair[i].nonce)|^{(4)}$
    $port \leftarrow$ lookup($pkt\_hdr$)
    trasmit($pkt, port$)
```

i. Check the source's timestamp in the received packet and compare it with the local time. If the difference is greater than 3 seconds, drop the packet, otherwise forward the packet according to Step (ii). Step (i) ensures that the source is not indicating false timestamps.
ii. Add a short nonce (4 bits) and a MAC (4 bits) at the corresponding AS-specific position in the header.
iii. Increment the $nextAS$ pointer.

Note that transit ASes do not need to perform destination-based key switching since they use their local secret to mark transit traffic. A non-cooperating AS ignores the FAIR header and forwards the packet according to the destination address. Procedure 2 summarizes these operations.

Processing of Inbound Packets: The destination performs the following data-plane operations:

i. Check the timestamp, similar to transit ASes.
ii. Detect sending policy violations per established communication channel. This is straightforward by using Token Bucket as a policer, given the CIR and CBS values.
iii. Verify MAC$_{K_{SD}}$ to ensure that the source's information has not been modified en route.

The destination stores the packet headers (network-layer headers and FAIR headers) as they contain potential proofs of misbehavior. Procedure 3 summarizes these steps.

2) Control Plane. We present a policy setup that introduces no latency in the data plane between the communicating end hosts of source and destination ASes. The setup is based on two concepts. First, ASes advertise their IP prefixes through BGP together with a default sending policy that is used until

Procedure 3: Processing of Inbound Packets

```
procedure RECEIVE($pkt\_hdr, fair$)
    ▷ $pkt\_hdr$ contains the network-layer packet header
    ▷ $fair[\,]$ is the FAIR header
    $diff \leftarrow |fair[0].time - \text{time}()|_{(16)}|$
    if $diff > 3$ and $diff < 2^{16} - 3$ then
        drop packet
    $icv \leftarrow \text{MAC}_{K_{SD}}(pkt\_hdr.payload\_len \,||\, fair[0].time \,||\, fair[0].seqno)|^{(8)}$
    if $icv \neq fair[0].icv$ then
        drop packet
```

source and destination ASes establish a new sending policy with different properties. Second, using mostly symmetric-key cryptography keeps the setup latency low. Specifically, only source and destination ASes sign the sending policy with their private keys, making the policy details provable and non-repudiable. Transit ASes insert MACs that remind them of being on the path of the communication channel. The combination of the aforementioned concepts allows end hosts to communicate without waiting for a sending policy setup and guarantees that the latency of the setup remains low.

First, we summarize all the information that is required and then we show how the policy is constructed.

- Current timestamp: inserted by the source AS, indicating the current time as the start for the communication channel.
- Expiration timestamp: inserted by the destination AS, indicating the end of the communication channel.
- Token Bucket properties: CIR and CBS values are inserted by the destination AS and specify the sending properties for the source (see Section II-A).
- FAIR-AS path: the source AS inserts the list of cooperating ASes on the path to the destination, which is known through the BGP advertisements.
- Autonomous System Numbers: each AS on the path inserts its own ASN that serves as an identifier.
- Signatures: source and destination ASes insert a signature over the policy details.

We provide more details about how this information is used.

The source AS creates a policy packet (P) and sends it to the destination AS. $P[0]$ corresponds to information inserted by the source AS and $P[n]$ to information inserted by the destination AS.

i. The source AS creates a policy packet P, with a timestamp indicating the start for the communication channel. Moreover, the source inscribes its Autonomous System Number ASN_0, the current time, the cooperating ASes on the path, and signs all the information with its private key PK_0^-. In particular, to avoid length-dependent security issues with signatures the hash of the information is signed [33].

$$\begin{aligned}
&P[0].asn \leftarrow ASN_0\\
&P[0].time \leftarrow \text{time}()\\
&P[0].path \leftarrow AS\_path\\
&P[0].sig \leftarrow \text{Sign}_0(\text{H}(P[0].asn \,||\, P[0].time \,||\, P[0].path))
\end{aligned}$$

ii. Each transit AS_i, $1 \leq i < n$, indicates its presence on the path. It adds its ASN_i and inserts a MAC over all the previous information. The MAC is computed with a long-term local secret ($\hat{K_i}$), known only to AS_i, that is used for control-plane operations.

$$\begin{aligned}
&P[i].asn \leftarrow ASN_i\\
&P[i].mac \leftarrow MAC_{\hat{K_i}}(\text{H}(P[0] \,||\, \cdots \,||\, P[i-1] \,||\, P[i].asn))
\end{aligned}$$

iii. The destination AS receives P and leverages the RPKI to verify the signature of the source AS. If verification succeeds, it fills in its ASN_n, the expiration time, and the Token Bucket values of CIR and CBS. The destination signs the contents of the final sending policy and sends it back to the source AS.

$$\begin{aligned}
&P[n].asn \leftarrow ASN_n\\
&P[n].expiration \leftarrow futureTime\\
&P[n].CIR \leftarrow CIR\\
&P[n].CBS \leftarrow CBS\\
&P[n].sig \leftarrow \text{Sign}_n(\text{H}(P[0] \,||\, \cdots \,||\, P[n-1] \,||\, P[n].asn\\
&\qquad ||\, P[n].expiration \,||\, P[n].CIR \,||\, P[n].CBS))
\end{aligned}$$

iv. Source and destination ASes use the RPKI and perform a non-interactive Diffie-Hellman key exchange to derive a shared key (K_{SD}) between them.

Note that transit ASes do not store information about the sending policy, and only indicate their presence in the communication channel. Moreover, only cooperating ASes indicate their presence in the communication channel. The source AS stores the final P for at least a period of $T_m = 12$ hours, as it is needed in the protest phase.

The signatures and MACs, by which each entity authenticates the information of all the previous entities, protect against path falsification attempts. A malicious entity cannot substitute the information inscribed by previous entities without invalidating the signatures or MACs. To avoid malicious entities from truncating on-path ASes, the source AS inserts the cooperating ASes on the path ($P[0].path$). In this way, on-path entities cannot truncate on-path ASes, as the source has indicated which ASes will cooperate. In addition, the source cannot lie and remove cooperating ASes from the indicated path, as these ASes will inscribe their information and reveal their support. Furthermore, the two timestamps indicate the validity period of the channel so that complaints are temporally confined.

D. Verifying Proofs of Misbehavior

In Section II-C we describe how the information in control and data plane is used to prove misbehavior. In this section, we describe the operations to examine a misbehavior report.

Recall that the information in the policy contains the transmission properties (CIR and CBS) for the communication channel. The data packet headers contain information for the actual transmission properties. The transit ASes examine the received information as follows.

i. ASes verify the signatures of the source and destination ASes in the policy packet, by obtaining the corresponding keys from the RPKI.
ii. ASes verify the 4-bit MAC that they inscribed in the header. If all verifications succeed, ASes proceed with Step (iii). MAC verification failures signal en-route misbehavior from a subsequent AS on the path from the source to the destination. In the next section, we analyze scenarios with on-path malicious ASes.
iii. The ASes check conformance to the Token Bucket properties by running Token Bucket as a policer and by using the timestamp and payload length information of the headers.

After the three-step procedure, the AS provides a signed admission or rejection for the misbehavior to the reporting AS. The destination AS collects the signed responses and sends them back to all ASes on the communication channel.

IV. Protocol Analysis

This section analyzes the security and scalability properties of FAIR.

A. *Security Aspects*

We analyze the security properties of short MACs and then describe to which extent FAIR is robust under two different threat models. We first consider a strong threat model in which all entities can be malicious. We then consider a second threat model that is slightly weaker, but specifically designed to address current attacks.

- Threat Model I: Misbehavior is provable at least to the benign cooperating ASes adjacent to the destination, under the strong threat model in which source, transit, and destination ASes can be malicious and collude.
- Threat Model II: Misbehavior is provable to *all* cooperating ASes on the path, under a weaker threat model in which transit ASes are not malicious.

Our goal to present a deployable high-performance system deals with the natural tradeoff between performance and security: some related approaches provide stronger security guarantees, but come at the cost of introducing considerable overhead. See Section VII for the details on related work.

1) On the use of short MACs. Before discussing the two threat models in detail, we evaluate the choice of short MACs. Specifically, we argue that a very short MAC is sufficient to provide accountability proofs in the context of flooding attacks. There are two important points to mention: i) The role of the MACs in the packet, as mentioned before, is only to provide a reminder to the transit ASes that they have forwarded the packet. In the context of flooding attacks, we care about an aggregate of packets and the collective proof that is constructed from this aggregate, rather than from single packets. ii) The secret keys used by other ASes are unknown to the attacker. This means that an attacker can at best randomly generate MACs without a means to check their validity.

The short length of the MACs does not prevent an attacker from generating valid MACs. However, for an 8-bit MAC, 99%

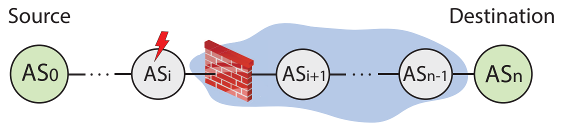

Fig. 4: FAIR operation with AS$_i$ being malicious.

of the generated MACs will not verify in the protest phase and the misbehavior will thus be detected.

Taking this approach to the limit, we could use 1-bit MACs for our purpose. An attacker would have a 50% probability to create a valid MAC. Thus, 50% of the crafted MACs would be invalid (compared to 99% previously) and the misbehavior is detected because of these invalid MACs. Our choice of the MAC lengths is based on engineering the protocol for high forwarding performance (byte aligned packet length), as we show in Section V-C.

2) Threat Model I. We first analyze the scenario of colluding ASes and then two scenarios with malicious transit ASes.

AS Collusion: In this scenario, a transit AS colludes with a malicious source AS to conceal an ongoing attack. The source is violating the sending policy and transit AS$_i$ (Figure 4) corrupts all the MACs of the previous ASes in the packet, causing verification failures when the destination protests to these ASes. Hence, the policy violation cannot be proven to these ASes. The first complaint round is successful only to the shaded ASes in Figure 4, as AS$_i$ cannot corrupt MACs of the subsequent ASes on the path. This limits the effectiveness of the proposal, however, successful complaints even to a few transit ASes yield benefits, as they can for instance install blocking filters closer to the source, as depicted in Figure 4.

The above scenario presents the worst case, in which a transit AS corrupts all previous MACs. If AS$_\mathrm{i}$ does not corrupt all the previous MACs, complaining is more effective since more ASes would acknowledge the attack. *Notice that the complaint is accepted at least by the benign cooperating ASes adjacent to the destination.* Hence, collusion with multiple ASes does not provide additional benefits to the source, as the effectiveness of the proposal depends only on the position of the malicious AS that is closer to the destination.

Packet Replay: In this scenario, we assume that a malicious transit AS forwards a packet multiple times to increase traffic and thus to blame an innocent source AS.

A packet replay is indicated through multiple occurrences of the same sequence numbers for a given timestamp. Furthermore, the clock deviation check does not allow an adversary to store packets and replay at a later time. The 24-bit sequence number suffices for more than $16 \cdot 10^6$ packets and the monotonically increasing values render multiple occurrences per timestamp suspicious. For example, a communication channel with a *CIR* value of 1 Gbps has an average packet-sending rate of 325 kpps for the average packet length of 413 bytes [34]. For an attack where each packet is replayed twice, there are on average 10^6 packets that belong to each slot of 1 second, but the 24-bit field suffices for more than $16 \cdot 10^6$ packets. Under normal operation each sequence number would show up only once, but multiple occurrences indicate a replay.

A high-sending-rate policy that uses up the available nonce space in the slot of 3 seconds would possibly allow an attacker to replay packets, but this is very unlikely: For the average packet size of 413 bytes it would require a communication channel of 17 Gbps for this to happen, which is an unrealistic value for a single channel. If such throughput values become reality in the future, increasing the sequence number length will solve the problem. For instance, 32 bits suffice for over 4 billion packets.

The nonce fields are used for detecting the adversary's location on the path: If AS_i (Figure 4) replays packets, then the combinations of sequence number and nonce field of only the first $i-1$ ASes occur multiple times. In other words, the location of the attacker can only be between AS_i and AS_{i+1}. The reason is twofold. First, non-cooperating ASes between AS_i and AS_{i+1} might replay packets. Second, the attacker (AS_i) might inscribe nonces in a way that puts the blame on the next AS (AS_{i+1}). Hence, the localization cannot identify the attack to a specific entity, but all ASes after the replaying AS become aware of the approximate location of the attack and can take action.

Note that we use sequence numbers and nonce fields to detect replay attacks in the protest phase, rather than to drop replayed data-plane traffic.

Packet Injection: In this attack, a transit AS attempts to craft fraudulent packets and inject them into the network. This attack is prevented thanks to the MACs inserted by the source and the transit ASes. Assuming that the adversary has not obtained the local secrets of the other entities, its probability of inserting only valid MACs is negligible, as discussed before. The verification failures of inserted invalid MACs will reveal the attack.

If AS_i (Figure 4) injects traffic, the subsequent ASes on the path insert their MACs as usual. These MACs will verify in the protest phase and hence the shaded ASes in Figure 4 acknowledge the violation, exactly as in the packet replay attack.

3) Threat Model II. Attacks usually originate from malicious or vulnerable end hosts inside the source AS; transit ASes usually have no incentive to collude with other ASes, nor to engage in malicious conduct, such as packet replay. The forwarding proof thus remains intact during transit and *all cooperating ASes on the path from the source to the destination acknowledge the attacks*.

Other Attacks. Here we describe some protocol manipulation attacks that are specific to FAIR. Since the destination uses the received packets as a proof of an attack, the source can craft timestamps in the packet, which together with the aggregate traffic size do not violate the policy. The clock deviation check protects against this, but allows the source to shift timestamps by one second, only once though. More specifically, the source can send excessive traffic in the slot of one second by putting the timestamp of the next second in some packets. In this way the maximum burst size violation for one time interval is not detected, but it restricts the traffic for the subsequent intervals, as it must be lower to conform to the policy's CBS value for the next intervals. A sending rate that exceeds the CIR value over any multiple of the time interval cannot be concealed. The Token Bucket properties in combination with the clock deviation check also protect from a coward attack [35]; in a coward attack the attacker scales down the intensity temporarily to avoid detection.

	Trace 1	Trace 2	Trace 3
Trace rate (Gbps)	1.63	3.72	3.57
IPv4 pkt. (bytes)	747 (99.95%)	920 (99.96%)	736 (99.88%)
IPv6 pkt. (bytes)	130 (0.05%)	342 (0.04%)	155 (0.12%)
BW overhead	**1.71%**	**1.39%**	**1.74%**

Fig. 5: Bandwidth overhead of FAIR for three backbone-link traces. The reported sizes are mean values and the parentheses show the percentage of traffic for each IP version.

Another general attack against accountability frameworks consists in falsely blaming benign entities. A malicious destination can try to convince transit ASes by providing multiple times the same packets as evidence of increased traffic. This is a variation of a replay attack and the sequence number and nonce fields prevent it. Crafting the timestamps will cause MAC verification failures and the transit ASes will not acknowledge the proof.

B. Scalability

We examine the scalability properties of FAIR in terms of bandwidth and storage overhead. Concerning the processing overhead, we provide a detailed evaluation in Section V.

1) Bandwidth Overhead. Our proposal comes at the cost of increased packet size. The source AS inscribes a constant amount of 7 bytes/packet and each transit AS adds another 1 byte. We envision a FAIR integration with the IP protocol and this would require two additional bytes per packet only in the case of IPv6 traffic (more details, also on IPv4, follow in Section V). To put this overhead into context, we analyze three 1-hour packet traces of OC-192 backbone links obtained from CAIDA [34]. We take a pessimistic approach on the AS-path length to quantify the overhead and assume it to be 5 hops.[2] Based on the number of packets in IPv4 and IPv6 and their ratio on the link, we calculate the link's overall bandwidth overhead. Figure 5 shows the properties of the traffic on the link and the overall overhead: the bandwidth overhead does not exceed 2%. This estimation assumes that the AS-path length is independent of the packet length distribution.

2) Storage Overhead. To provide a scalable framework, our goal is to reduce the amount of state stored at the forwarding devices of cooperating ASes. Source and transit ASes do not need to store data-plane related information. The source stores one policy packet and a shared key K_{SD} (16 bytes) per communication channel. The total number of ASes in the Internet is less than 50,000 [36], which means minimal overhead (800 kB) even if there is a communication channel with every other AS.

Furthermore, the transit ASes store only local secret keys (independent of any communication channel). As noted in Section IV-A, there is no strict requirement on the frequency of changing keys, however, the previous keys are kept to verify MACs that were computed earlier. According to the protocol,

[2]RIPE Labs report an average length of 3.9 hops for IPv4 and 3.5 hops for IPv6 [32].

a cooperating AS accepts and examines incoming proofs up to a period of $T_m = 12$ hours in the past. Hence, the storage overhead depends on the frequency with which the AS changes its keys within the 12-hour frame. For example, a transit AS that changes its local keys (K_i, $\hat{K_i}$) every minute requires a storage capacity of 250 kB for the 12-hour period.

The most significant storage overhead occurs for the destination AS when storing data packet headers as a proof of source misbehavior. The destination can provide the proof to the transit ASes up to 12 hours after it received the packets. For a destination AS that stores the IP and FAIR packet headers of the 1-hour link traces in Figure 5, the storage requirement is 30.2, 56, and 67.3 GB, respectively. For this calculation, we assume again an AS-path length of 5 hops and took into consideration the different overhead of the IPv6 header (40 bytes) and the IPv4 header (20 bytes), on top of the FAIR header overhead.

Note that the considerable storage overhead is shifted to the destination AS since it is in the destination's interest to be protected from flooding attacks; thus having forwarding ASes store the packets would distribute the storage overhead in an unfair manner. Moreover, to further decrease the overhead, destination ASes store only packet headers. Also, the destination can choose when to protest about a violation, hence it does not have to store headers for 12 hours and can regulate the storage requirement. In addition, ASes can store compressed proofs of misbehavior only for the violated time periods instead of storing the whole set of packets of the communication channel.

V. Implementation and Evaluation

We describe our protocol in the context of today's Internet, implement a software switch prototype, and evaluate performance on a server and a desktop machine.

A. Integration with IP

We analyze the deployment of FAIR with IP. IPv6 allows a straightforward and elegant implementation by using Extension Headers (EHs) [37]. IPv6 Extension Headers encode optional IP-layer information in headers that are placed after the regular IPv6 header. They make the protocol extensible by allowing support for security, mobility, and other services.

The IPv6 specification [37] defines some default EHs for additional network-layer services and leaves space for new EHs. To implement FAIR, we define a new EH that is processed only by egress border routers of cooperating ASes. According to the specification, the Hop-by-Hop EH is the only EH that *must* be processed by all network devices, whereas other EHs are inspected only by devices configured for certain services. This feature allows ISPs to adopt FAIR in an incrementally deployable fashion without breaking legacy IPv6 traffic. Figure 6 shows a regular IPv6 header together with the FAIR extension. The FAIR EH is placed after the regular IPv6 header or after the Hop-by-Hop EH (if present), as the IPv6 specification commands. The `Next Header` field (whether in the regular header or in a preceding EH) points to the start of the FAIR EH. The content of our EH is what Section III describes and Figure 2 depicts. To make FAIR compatible with IPv6, two additional fields are required: a

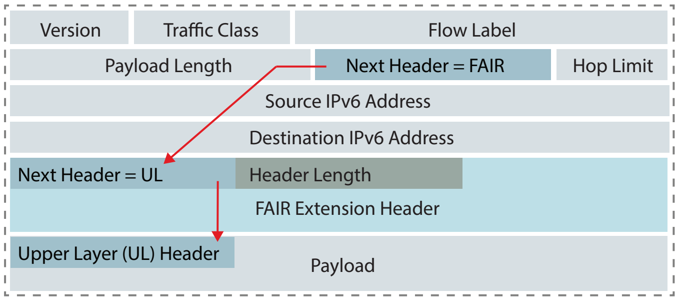

Fig. 6: IPv6 packet with FAIR Extension Header.

pointer (8 bits) that points to the next EH or to an Upper Layer (UL) protocol, and a `Header Length` field (8 bits) that indicates the length of the EH. This translates into an additional overhead of 2 bytes.

Extension Headers are considered an intrinsic part of IPv6 and the way they are processed by network devices can harm forwarding performance. However, IPv6 provides an elegant deployment path due to EHs. This feature is not supported by IPv4 and a workaround for IPv4 is necessary.

IPv4 has inherent limitations with regard to extensibility which complicates deployment. The FAIR header can be implemented as a "shim" layer between the IPv4 header and the transport protocol. The border routers of source ASes insert the FAIR header after the IPv4 header; border routers of transit ASes locate and process the FAIR header, as it starts 20 bytes after the IPv4 header; and the border routers of destination ASes store and remove the FAIR header before forwarding the packet to the destination host. Shim-layer approaches typically cause problems due to middleboxes in the source and/or destination ASes [38]. However, note that the FAIR header is not visible inside those domains, alleviating such concerns.

B. Software Switch Prototype

To test the practicality of our proposal, we implement the required functionality in software. We recognize a resurgence of interest in software switches thanks to their flexibility and programmability at low procurement and operational costs [39–41]. Furthermore, recent advances in the software-switching field demonstrate that these advantages do not come at the cost of performance, which has traditionally been the Achilles' heel of software switches. We use the Intel Data Plane Development Kit (DPDK) [42] as the packet I/O engine and take advantage of the Intel AES-NI [43].

The **Intel DPDK** is a high-performance packet I/O engine that provides flexibility and programmability, allowing packet processing in user space. DPDK uses polling to avoid the overhead of unnecessary interrupts. It provides optimized Network Interface Card (NIC) drivers that map packet buffers directly in user space to avoid redundant memory accesses (zero copy). We choose DPDK for our development platform as it efficiently performs packet I/O and allows us to focus on the FAIR EH processing.

The **Intel AES-NI** is a recent instruction set that uses hardware support to speed up encryption and decryption of

AES operations. Intel reports a performance of 2.01 Cycles Per Byte (CPB) for a 16-byte block AES encryption on an Intel Westmere running at 2.67 GHz [43].

We describe the implementation of the necessary components for FAIR. To construct the required MACs, we use the Cipher Block Chaining mode (CBC-MAC) with AES as the underlying block cipher. The CBC-MAC encryption of a plaintext block depends on the encryption of the previous block; the output is the final block. The value for the Initialization Vector (IV) is 0. The size of both input blocks and the output block is 128 bits (16 bytes). The input length to the CBC-MAC is fixed and independent of the AS path length[3]. Also, the input fits in one block (less than 16 bytes). Furthermore, the input length of the MACs in the control plane is fixed as well. We use 128-bit encryption keys and keep only the required number of bits from the output, as specified in Section III.

The source AS of the outgoing traffic has to look up the shared key with the destination (K_{SD}) and the current packet count for the communication channel, as it is used for the sequence number ($seqno$). The source uses the shared key with the destination in order to compute the MAC. To implement these functionalities at line rate, we extend the Forwarding Information Base (FIB) to contain not only the egress interface, but also the shared symmetric key with the destination and the current value for the sequence number.

This increases the size of the FIB, but it still fits in todays SRAM caches, avoiding access to the substantially slower DRAM. The size of the extended FIB for today's IPv4 BGP routing table sizes is around 12 MB [45] and for IPv6 around 1 MB [45], which is lower than SRAM sizes even on commodity hardware, as we show in our evaluation. In addition, the increase in length for each FIB entry does not degrade forwarding performance since each FIB entry fits into the typical cache line of 64 bytes. Even in case of IPv6 addresses, where each entry requires 36 bytes (16-byte destination address, 16-byte symmetric key, 3-byte sequence number, and 1-byte output interface).

To generate randomness for the nonce and to mark fields at line rate, we need an efficient pseudorandom number generator (PRNG). We implement a thread-safe, multicore version of the Linear Congruential Generator (LCG) that meets our performance requirements. Modern CPUs come with Digital RNG (DRNG) hardware implementations [46] that can speed up this process significantly [47]. Unfortunately, our CPUs lack this feature. Furthermore, each CPU core has an AES hardware unit. We assign each core to handle one port, taking advantage of the processing power of today's multicore systems. For the timestamp, we use the least significant bits (LSB) of the Unix time.

We bring these components together on two different machines: a commodity server and a low-end desktop. The server has a non-uniform memory access (NUMA) architecture with two Intel Xeon E5-2680 CPUs that communicate over two QPI links. Moreover, each NUMA node is equipped with four banks of 16 GB DDR3 RAM. In total, we have 6 dual-port 10 GbE NICs (PCIe Gen2 x8) that can provide a maximum capacity of 120 Gbps. The total cost of this setup is around $\$7,000$. Figure 7 summarizes the hardware specification of the server machine.

[3]CBC-MAC is insecure for variable-length messages [44].

Item	Model Name	Qty	Unit price
Board	Intel S2600GZ (2 sockets)	1	$670
CPU	Intel Xeon E5-2680 (8 cores, 2.7 GHz)	2	$1,727
RAM	Kingston DDR3 4 GB (1,333 MHz)	8	$38
NICs	Intel 82599EB X520-DA2 10 GbE	6	$450

Fig. 7: Specification of utilized Server Hardware.

Item	Model Name	Qty	Unit price
CPU	Intel Core i5-3470S (4 cores, 2.9 GHz)	1	$170
RAM	Hynix DDR3 4 GB (1,600 MHz)	1	$45
NICs	Intel 82599EB X520-DA2 10 GbE	1	$450

Fig. 8: Specification of utilized Desktop Hardware.

The desktop machine is a Lenovo ThinkCentre Edge 3494AZG with an Intel Core i5-3470S CPU with one dual-port 10 GbE NIC (PCIe Gen2 x8) and a total cost of $\$1,200$. Figure 8 shows the hardware specification of the desktop machine.

C. Switch Prototype Evaluation

We evaluate the switching performance of both machines and demonstrate that the EH processing incurs minimal computational overhead even for low-end hardware.

In the experiments, we emulate traffic flows originated by a source AS and evaluate the performance of a FAIR-enabled border router. We evaluate the worst case, and thus we use IPv6 that is slower than IPv4 because the Forwarding Information Base (FIB) entry is longer than for IPv4; we have observed the same forwarding performance also for IPv4 traffic. Moreover, we specify random destination addresses for the generated flows, eliminating spatiotemporal locality for cache accesses. Using random destination addresses captures any performance degradation due to key switching with different destination ASes. To generate traffic, we use Spirent SPT-N4U-220 as our packet generator. The table lookup is performed by an implementation of DIR-24-8-BASIC [48] for IPv6 addresses. We generate the FIB from a BGP routing table snapshot (November 2014) from RIPE RIS, with 18k unique IPv6 prefixes [45].

First, we evaluate the performance of a single 10G port for three packet sizes; then we enable all ports. Finally, we evaluate performance with all ports enabled and for varying packet sizes. All the experiments are conducted on the server and the desktop platforms.

1) Single-port experiment. First, we test the switching performance of one port for three packet sizes: 68, 128, and 1024 bytes. Minimum-sized packets, 68 bytes, translate to a higher packet rate and are the worst case for the EH processing. The minimum length for IPv6 packets with the FAIR EH is 68 bytes (instead of 64) due to the additional information. Figure 9 shows the switching performance for the server and the desktop platform.

The highest packet rates for the three packet sizes are 14.20 Mpps, 8.45 Mpps, and 1.20 Mpps on a 10 GbE link;

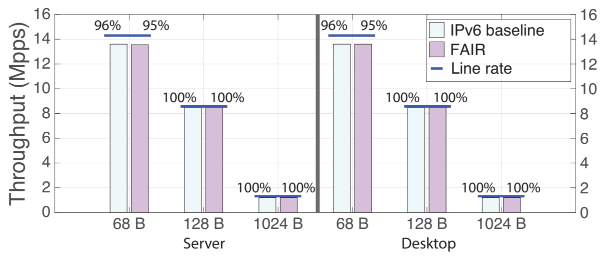

Fig. 9: Switching performance of the server and the desktop for one port activated, and 68, 128, and 1024-byte packets.

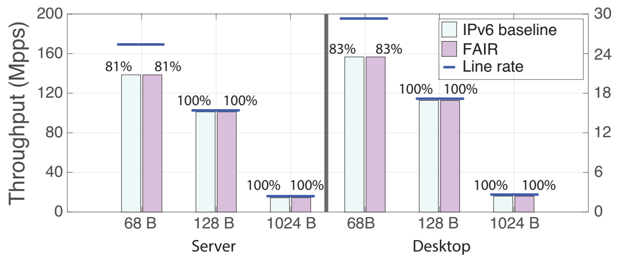

Fig. 10: Switching performance of the server and the desktop for all ports activated, and 68, 128, and 1024-byte packets.

we refer to these values as the line-rate performance. The baseline for the experiments is the switching performance of legacy IPv6 traffic (only table lookup and forwarding). The figure shows that the EH processing degrades performance by only 1% for minimum-sized packets on both machines. The figure also shows the line-rate performance (blue line) and the minimal baseline degradation due to the table lookup and the high packet rate for the 68-byte case. For the longer packet sizes, the switching performance reaches the line rate on both machines. The single-port experiment demonstrates that switching performance is close to optimal for one port, even on low-end hardware. Next, we increase the switching load.

2) All-ports experiment. To demonstrate that the FAIR EH processing scales for increasing packet rates, we activate all ports; each port is served by a different CPU core. Again we use the same three packet sizes. Figure 10 shows the results.

We use a different scale in the figure for the two machines, since they accommodate a different number of ports. The packet line rates for the server (12 ports) and the three packet sizes are 170.4 Mpps, 101.4 Mpps, and 14.4 Mpps, respectively. The packet line rates for the desktop (2 ports) and the three packet sizes are 28.40 Mpps, 16.90 Mpps, and 2.40 Mpps, respectively. We see that throughput scales for multiple ports and FAIR switches at baseline performance for the three packet sizes, on both machines. The experiment demonstrates how switching performance scales for increasing packet rates, even for the low-end hardware. However, we notice a higher baseline degradation for 68-byte packets: in the one-port experiment, the switching performance was at 96% of the line rate, whereas now it is around 80%. The explanation is that our I/O subsystem hits a bottleneck when both ports of a NIC receive packets at the maximum packet rate. The bottleneck exists irrespective of FAIR: the PCIe Gen2 x8 interface of our NICs cannot sustain this packet rate when both ports are active. The packet rate of each port is capped at 11.55 Mpps. Cuckooswitch [39] uses the same NICs and reports the same limitation.

3) CPU as the bottleneck. To bypass the I/O bottleneck and stress the limits of the CPU, we assign the traffic from two ports of different NICs to one core; this makes the CPU the throughput bottleneck. For minimum-sized packets, the CPU handles 21.62 Mpps out of the maximum 28.40 Mpps. Hence, one CPU core can process traffic from more than one 10 GbE port that receives packets at the maximum packet rate.

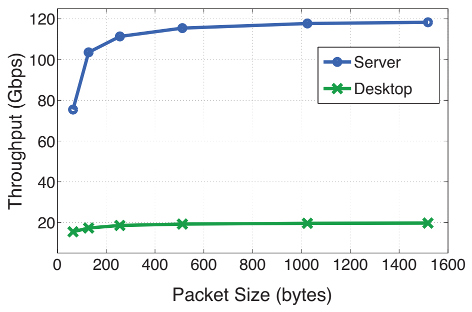

Fig. 11: Switching performance for all ports.

Next, we show that for increasing packet sizes, FAIR saturates line-rate bandwidth and achieves 120 Gbps and 20 Gbps for the server and desktop respectively. Figure 11 shows the throughput for 68, 128, 256, 512, 1024, and 1518-byte packets. We omit the line-rate line; for all measurements — except the 68 byte packet — it is identical to the drawn lines. Hence, as we increase the packet size and the packet rate drops, IPv6 baseline and FAIR performance is at 100% line rate.

VI. Protection from DDoS Attacks

FAIR, as an accountability framework, does not provide active protection from attacks, as it does not enforce specific behavior when an attack is detected. This section describes a more radical application of FAIR that enforces and pushes higher security standards to the edge of the Internet. Furthermore, the section illustrates how FAIR can be combined with active defense mechanisms.

A. *Suspicious Bit*

The April Fool's proposal of the "evil bit" [20] describes a security mechanism from an idealist's point of view: data packets carry a security flag – the *evil* bit – to indicate malicious intent; the flag is set by the malicious senders themselves.

We propose a more realistic security mechanism, the *suspicious* bit that is set by transit ASes to indicate suspicious traffic. With such a mechanism in place, the traffic itself becomes the indicator of possibly malicious behavior and incentivizes transit ASes to take action. For instance, an AS can drop or deprioritize suspicious traffic in case of congestion, ensuring better service for its benign customers. In addition, flagging traffic due to an attack on one victim provides protection to other potential victims as well.

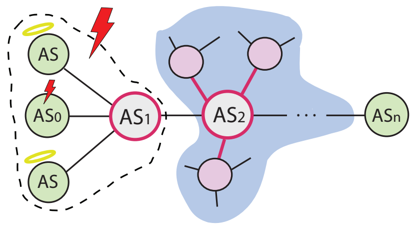

Fig. 12: The suspicious bit identifies traffic from a portion of the network with poor security practices.

An immediate question is how ISPs distinguish benign from suspicious ASes in order to flag their traffic. We leverage FAIR as a building block to address this question. FAIR's initial sending policy negotiation provides a clear line for detection of misbehavior; the FAIR header in the data packets provides the corresponding accountable proofs of misbehavior.

Another question is how ISPs are incentivized to flag their misbehaving customers. The answer to this question lies in the competitive environment in the Internet ecosystem. Recall that FAIR's accountable proof of misbehavior is received by all on-path ASes. If an ISP does not flag its provably malicious transit traffic, then the next AS on the path will flag *all* of the traffic of the previous AS. We believe that the threat of collateral damage and the harsh competitive Internet market pushes ISPs to mark their customers' traffic. If innocent customers experience packet drop because of their ISPs' poor security practices, they have an incentive to switch to a more reliable ISP, if possible.

We emphasize that ISPs do not have to drop suspicious traffic right away for two reasons. First, the suspicious bit indicates only that the traffic is suspicious (not necessarily malicious) and thus gives incentives to take action under certain conditions (e.g., drop it in case of congestion). Second, under the strong threat model, an adversary could set the bit for legitimate traffic to make another ISP drop the traffic. Consequently, setting the suspicious bit for legitimate traffic would not be a useful attack strategy. In addition, today's Internet is opaque to loss anyway [49], and hence the adversary can directly drop the traffic and evade detection.

We demonstrate the suspicious bit application by means of Figure 12. The illustrated network topology shows malicious AS_0 violating the sending policy negotiated with benign AS_n. AS_1 is the ISP of the malicious AS_0 and other benign ASes. It hence provides transit to more than a single customer. Assume that AS_1 has received a proof of misbehavior for AS_0: AS_n has reported malicious traffic to AS_1. In the ideal case, AS_1 would mark traffic from AS_0 as suspicious, warning other entities in the network. If, however, AS_1 does not mark the suspicious traffic, then AS_2 will mark *all* the traffic from AS_1 as suspicious.

This overstatement, however, means that also traffic from the *benign* customers of AS_1 gets flagged as suspicious, which will lead to collateral damage if a downstream ISP decides to drop traffic. By flagging traffic, AS_2 informs other entities in the network (shaded part) that some portion of the network (dashed part) might be misbehaving. This practice will incentivize AS_1 to behave correctly and to flag the traffic of its misbehaving clients, thereby protecting its benign clients. As a consequence, the stub ASes are pushed to deal with their internal security issues (e.g., botnets inside an AS or misconfigured services) to protect the innocent flows from being dropped.

Forwarding with the Suspicious Bit: We show the information and data structures when forwarding traffic under the SB application.

- Suspicious Bit (sb): the SB flag, used to mark a packet as suspicious, is the most significant bit of the $nextAS$ pointer in the FAIR header. This means that routers will check and update the 7 least significant bits of the pointer, which suffice to encode AS-paths of length up to 128 hops.
- Suspicious Sources ($sus\_sources$): set of addresses for which the AS has acknowledged the violation.
- Suspicious Ports (sus_ports): set of the switch's ports that receive traffic from an insecure part of the network. We refer remaining ports of the switch as non-suspicious.

In the following, we describe how this information is used to realize the suspicious bit application. Note that the SB does not enforce a specific action, hence the transit AS can forward, drop, or delay traffic based on its traffic engineering and security policies. Procedure 4 provides the pseudocode for traffic forwarding with the suspicious bit.

- If incoming traffic arrives at a non-suspicious port:
  - if the SB is set then forward/drop/delay traffic.
  - if the SB is not set and the source address belongs to the suspicious sources then add the port to the suspicious ports. Set the SB and forward/drop/delay traffic.
- If incoming traffic arrives at a suspicious port:
  - if the SB is set, remove the incoming port from the suspicious ports. In this way if previous ASes that did not flag traffic start flagging, their whole traffic is not flagged as suspicious anymore. Then forward the traffic.
  - if the SB is not set, then set the SB. Then forward/drop/delay.

B. Active Defense

We describe how forwarding accountability serves as a building block for active DDoS defense. Transit ISPs can simply drop traffic from malicious ASes, providing a primitive DDoS defense. However, accountable proof of misbehavior can be combined seamlessly with more sophisticated protection schemes.

Filtering defense proposals (e.g., StopIt [14], AITF [15], and Pushback [16]) demonstrate the effectiveness of a distributed and cooperative approach to control certain traffic flows by asking upstream routers to install filters. These approaches assume that upstream routers are willing to install such filters. However, at the inter-domain level this is a strong assumption.

ISPs are harsh competitors and are mutually distrusted entities. In addition, ISPs earn revenue by forwarding traffic, Procedure 4: Forwarding packets in the SB application

```
procedure FORWARD($pkt\_hdr, fair, port\_in$)
    ▷ $pkt\_hdr$ contains the network-layer packet header
    ▷ $fair$ is the FAIR header
    ▷ $port\_in$ is the ingress port of the packet
    if $port\_in \notin sus\_ports$ then
        if $fair.sb$ then forward/drop/delay traffic
        else
            if $pkt\_hdr.src\_addr \in sus\_sources$ then
                $sus\_ports \leftarrow sus\_ports \cup \{port\_in\}$
                $fair.sb \leftarrow 1$
                forward/drop/delay traffic
            else
                forward traffic
    else
        if $fair.sb$ then
            $sus\_ports \leftarrow sus\_ports - \{port\_in\}$
            forward traffic
        else
            $fair.sb \leftarrow 1$
            forward/drop/delay traffic
```

regardless of the intent of the traffic. Furthermore, filtering resources at forwarding devices are limited and should be used cautiously. Hence, spending filtering resources for targets outside the AS boundaries is an assumption that does not hold. StopIt [14] recognizes this fact for inter-domain filtering requests and leverages shared keys to authenticate such requests. However, no filtering proposal obtains proof of misbehavior in order to install such filters. Malicious ASes could try to exhaust filtering resources of other ASes.

FAIR allows an AS to provide misbehavior proof to other ASes and convince them to install filters. Furthermore, accountability can lead to novel contractual regimes and SLAs that formally describe cooperative mechanisms to address the flooding attacks.

We discuss the deployment and operation of FAIR. The prominent advantage of FAIR is founded on the fact that collateral damage can be leveraged to push ISPs to enforce higher security standards, e.g., to deal with internal security threats such as botnets or vulnerable components. Collateral damage mainly stems from today's Internet architecture, and specifically from its lack of accountability. In particular, in distributed attacks, the misbehaving source end hosts cannot be identified.

FAIR identifies such malicious sources at the AS granularity with the consequence that also innocent flows get classified as malicious. Clearly, harming innocent flows is undesirable, but provable AS misbehavior gives incentives for ISPs to take action against such malicious traffic (e.g., deprioritize or drop it). This holds the whole AS accountable for misbehavior and puts it under pressure to deal with its security issues, rather than delegating flooding protection to the victim. Hence, provable misbehavior turns collateral damage on its head by using innocent flows as a way to pressure ASes to deal with their security issues.

C. Deployment Path

FAIR is deployable in the context of today's Internet as it does not require architectural changes. More precisely, FAIR is compatible with today's protocols and especially with IPv6 extension headers, which were designed for deploying novel protocols. The introduced overhead, although not negligible, is within reach of today's processing and networking capabilities. In addition, given that source and destination ASes set up a sending policy, the destination can protest and prove misbehavior even if only one transit AS supports FAIR. Thus, ASes can deploy FAIR independently without global coordination.

On the downside, forwarding devices on the data path will need to support additional processing mechanisms, which translate to upgrades and costs. Furthermore, the considerable storage overhead for destination ASes can further increase operational costs. Finally, the requirement for a policy construction that defines the characteristics of the transmission constitutes a deviation from today's communication model.

D. Operational Assumptions

In the high-level overview of FAIR (Section II), we presented a router-level communication model between the source and destination AS in which we assumed that all traffic flows originated by the source AS follow the same AS-level path towards the destination. We relax this assumption of a line topology, as this model does not reflect reality: each border router decides independently on the next AS hop. Moreover, the interaction of inter-domain routing and intra-domain traffic engineering (e.g., load balancing) leads to different AS-level paths between the source and destination ASes. Therefore, in FAIR, a communication channel is identified by the AS path and not by the source-destination AS tuple.

Furthermore, two ASes can peer at multiple Points of Presence. Consequently, the source AS might have to coordinate the sending rates if there are multiple peering points with the next AS. Readily available approaches deal with such traffic engineering tasks: Segment Routing Centralized Egress Peer Engineering developed by Cisco [50] and Intelligent Route Service Control Point solutions [51] are such examples.

Routing instability that forces source and destination to reestablish a communication channel over a new path is not a notable concern. Studies show that the majority of network routes are stable from tens of minutes to days [52, 53]. Despite ISPs' traffic engineering and the existence of short-lived routes, long-lived routes are used 96% of the time [52].

Furthermore, today's border routers are not required to perform cryptographic operations on data-plane traffic. However, the recent advances in cryptographic engines, such as Intel AES-NI [43], allow efficient cryptographic operations even for commodity machines, as we have demonstrated in Section V-C.

Moreover, schemes that increase the packet length (the border router of the source AS adds the FAIR header) need to take into account correct MTU discovery. In case a large packet requires fragmentation, the border router of the source AS can respond with an MTU size small enough, so that the FAIR header can be added without concerns.

E. Security Concerns

In this paper, we focused on the security properties of the accountability framework and not on other security aspects (such as source accountability or flooding attacks on the channel setup). Source address spoofing is a well-known and studied problem with best current security practices (BCP 38/84 [54, 55]) that should be followed by administrators. Denial-of-Capability (DoC) attacks – flooding the request channel of capability defense systems – have been demonstrated along with proposals for defense [11, 12], which can be used as protection from flooding the FAIR setup channel. We stress that our key ideas are compatible with other future Internet proposals that address natively the aforementioned security concerns [25, 26].

VII. Related Work

We describe some major accountability and DDoS defense schemes; comprehensive surveys about DDoS defense can be found in Zargar et al. [56] and in Mirkovic et al. [57].

Accountability mechanisms are building blocks to hinder DDoS attacks, rather than active defense mechanisms. For example, **AIP** [7] is a network architecture based on accountability, with a two-level flat addressing structure that allows for using self-certifying addresses (the hash of the corresponding public keys). **IPA** [58] is a more lightweight approach that binds an IP prefix to the public key of an AS by leveraging the DNSSEC infrastructure. The secured bindings are piggybacked in BGP messages and get distributed in a protocol-compliant and incrementally-deployed way. **Passport** [8] is a network-layer source authentication system that authenticates the source of a packet to the granularity of the origin AS. Symmetric key cryptography is used and packets are checked only at administrative boundaries. Using accountable source addresses as a building block, additional defense schemes are proposed. For example, a shut-off protocol is proposed [7], where a host can instruct the network interface of an attacker to stop packet transmission. However, this pushes DDoS defense to the hosts, assuming that all hosts recognize such a shut-off protocol.

Simon et al. propose AS-based accountability as a cost-effective DDoS defense [59]. Moreover, the authors propose an evil bit in the packet headers. The proposal works for a group of participating ASes, assuming pairwise and transitive trust between them. The evil bit is set whenever traffic enters from outside the island of the participating ASes. However, the inferred threat model is weak, since a single compromised AS inside the group of participating ASes limits the effectiveness of the proposal. In addition, the system introduces considerable upgrades in terms of infrastructure and requires new Customer Relationship Management (CRM) systems.

Other accountability schemes used for debugging and forensics introduce prohibitive overhead for deployment in the data plane. SNP [60], PeerReview [61], and NetReview [62] keep detailed logs of exchanged messages and introduce substantial overhead in terms of processing, storage, and bandwidth.

An alternative approach to identify the source of an attack is to identify the path(s) traversed by malicious traffic. In **IP traceback** [21], downstream routers probabilistically mark packets with partial path information. The victims combine the partial path information in the packets to reconstruct the path(s) to the source(s) of the attack. The proposal yields high computational overhead for path reconstruction at the victims and a high false positive rate even for small scale DDoS attacks [63]. In addition, IP traceback operates under a weak threat model, in which downstream routers need to be trusted. Incremental proposals optimize the computational overhead and operate under a stronger threat model that includes malicious routers [63]. **Hop-Count Filtering** [64] is a host-based approach that discards spoofed DDoS traffic. The main idea is that the only IP header information that cannot be influenced by an attacker is the TTL field. Hence, spoofed IP packets will most probably have inconsistent hop-count values with the IP addresses being spoofed. FAIR borrows ideas from these schemes, as the packets contain proofs of misbehavior if the source violates the acknowledged traffic profile. The destination then sends the proofs *back* to the corresponding ASes to prove the misbehavior.

There are two main approaches for *active* defenses against DDoS attacks: capabilities and filtering. Capability proposals [10–12, 65] let the destination explicitly authorize traffic that it desires to receive. Our approach is inspired by capability schemes — not for proving traffic legitimacy, but for collecting and providing proofs to each transit AS on the path. The first challenge for a victim is to distinguish between malicious and benign traffic sources [65]. Benign traffic sources get short-term authorizations – capabilities – from the destinations and put them into the packets, so that the legitimacy of traffic can be verified. Capability proposals introduce considerable complexity and are susceptible to DoC attacks [10]. To address DoC, **TVA** [11] tags each packet with the identifier of the ingress point to an AS and fair-queues packets at each router according to this identifier. **Portcullis** [12] uses puzzles (computational proofs of work) to provide fair sharing of the request channel. **NetFence** [66] is a hybrid system and introduces a secure congestion policy feedback combined with elements from capability-based systems. Most capability proposals assume a mechanism that distinguishes malicious from benign traffic and the effectiveness of these proposals is, at most, as good as this assumed mechanism. In FAIR, we use a traffic profile that draws a clear line between malicious and benign behavior, and use the proofs in the packets to push the edge ASes to address their security problems.

The second class of active DDoS defense mechanisms, filtering proposals, relies on stopping malicious flows in the network before reaching the victim. **StopIt** [14] uses a closed-control and open-service architecture to defend from attacks that prevent filter installation. End hosts can send StopIt requests only to their access routers and each AS has a StopIt server that handles StopIt requests. **AITF** [15] installs filters in routers as close as possible to the attacking sources, rather than in backbone routers. **Pushback** [16] detects a malicious traffic aggregate and controls it at a single router and in a cooperative manner by asking upstream providers to stop the malicious aggregate. Such filtering schemes assume cooperation among ISPs and that ISPs are willing to provide some of their filtering resources to protect remote victims. However, this is an unrealistic assumption in today's competitive Internet ecosystem and we consider the accountable proof of misbehavior as a way to convince ISPs to install filters. Alternatively, such proof can lead to new contracts among ISPs with regard to security.

VIII. Conclusion

This paper has presented FAIR, an attempt to answer the question on how to incentivize ISPs to adopt stricter security policies and thereby to secure the insecure edge of the Internet where most of today's security problems are rooted.

FAIR leverages forwarding accountability to prove to transit ISPs on the path from the source to the destination that they have forwarded (malicious) traffic. Using FAIR's accountable proof of misbehavior, we have presented an application – the suspicious bit – that incentivizes ISPs to mark traffic from their suspicious customers as such and thereby inform other entities in the network. FAIR comes with less than 2% bandwidth overhead and without any storage overhead for the transit ISPs. Furthermore, FAIR is incrementally deployable in today's Internet, and it gives incentives for early adoption.

We have implemented a FAIR software switch that processes packets at the line rate of 120 Gbps, and forwards 140M minimum-sized packets per second.

References

[1] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir, "Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks," in *Proceedings of the 2014 Internet Measurement Conference*, 2014.

[2] "DDoS Attack Hits 400 Gbit/s, Breaks Record," "http://ubm.io/1HOGEpr", 2014.

[3] "Q4 2014 State of the Internet Security Report," "http://bit.ly/1QIBg9H", May 2015.

[4] "China's Man-on-the-Side Attack on GitHub," "http://bit.ly/1EnxqMU", Mar. 2015.

[5] "Dissection of 'itsoknoproblembro', the DDoS tool that shook the banking world," "http://bit.ly/1HOGCO5", Jan. 2013.

[6] "Millions of websites hit by Drupal hack attack," "http://bbc.in/1DtMJQd", Oct. 2014.

[7] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker, "Accountable Internet Protocol (AIP)," in *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2008.

[8] X. Liu, A. Li, X. Yang, and D. Wetherall, "Passport: Secure and Adoptable Source Authentication," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008.

[9] "Cloudflare Features and Pricing," "https://www.cloudflare.com/plans", 2015.

[10] A. Yaar, A. Perrig, and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," in *Proceedings of the 25th IEEE Symposium on Security and Privacy*, 2004.

[11] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting Network Architecture," in *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2005.

[12] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting Connection Setup from Denial-of-capability Attacks," in *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2007.

[13] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet Denial-of-Service with Capabilities," *SIGCOMM Computer Communication Review*, Jan. 2004.

[14] X. Liu, X. Yang, and Y. Lu, "To Filter or to Authorize: Network-layer DoS Defense Against Multimillion-node Botnets," in *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2008.

[15] K. Argyraki and D. R. Cheriton, "Active Internet Traffic Filtering: Real-time Response to Denial-of-service Attacks," in *Proceedings of the USENIX Annual Technical Conference*, 2005.

[16] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," *SIGCOMM Computer Communication Review*, Jul. 2002.

[17] "Routing Resilience Manifesto," "https://www.routingmanifesto.org/", May 2015.

[18] P. Gill, M. Schapira, and S. Goldberg, "A Survey of Interdomain Routing Policies," *SIGCOMM Computer Communication Review*, Dec. 2013.

[19] Federal Communications Commission, "Open Internet Order," "http://www.fcc.gov/openinternet", Feb. 2015.

[20] S. Bellovin, "The Security Flag in the IPv4 Header," RFC 3514 (Informational), Internet Engineering Task Force, Apr. 2003.

[21] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," in *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2000.

[22] "Nmap Network Scanning," "http://bit.ly/1DPn4BK", May 2015.

[23] Cisco, "Cisco Policing and Shaping Overview," "http://bit.ly/1HOHr9V", May 2015.

[24] I. Castro, A. Panda, B. Raghavan, S. Shenker, and S. Gorinsky, "Route Bazaar: Automatic Interdomain Contract Negotiation," in *15th Workshop on Hot Topics in Operating Systems (HotOS XV)*, 2015.

[25] D. Barrera, R. M. Reischuk, P. Szalachowski, and A. Perrig, "SCION Five Years Later: Revisiting Scalability, Control, and Isolation on Next-Generation Networks," *arXiv/1508.01651*, Aug. 2015.

[26] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION: Scalability, Control, and Isolation on Next-Generation Networks," in *Proceedings of the 31st IEEE Symposium on Security and Privacy*, 2011.

[27] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271 (Draft Standard), Internet Engineering Task Force, Jan. 2006.

[28] ARIN, "Resource Public Key Infrastructure," "http://bit.ly/1EJCQoT".

[29] A. Aziz and M. Patterson, "Design and Implementation of SKIP," in *Proceedings of INET*, 1995.

[30] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008.

[31] P. H. Dana, "Global Positioning System (GPS) Time Dissemination for Real-time Applications," *Real-Time Syst.*, Jan. 1997.

[32] RIPE Labs, "Update on AS Path Lengths Over Time," "https://labs.ripe.net/Members/mirjam/update-on-as-path-lengths-over-time".

[33] J.-S. Coron, "On the Exact Security of Full Domain Hash," in *Proceedings of the 20th Conference on Advances in Cryptology*, 2000.

[34] "CAIDA: Center for Applied Internet Data Analysis," "http://www.caida.org".

[35] B. Liu, J. T. Chiang, J. J. Haas, and Y.-C. Hu, "Coward attacks in vehicular networks," *Mobile Computing. Communications Review*, Dec. 2010.

[36] "CIDR Report for July," "http://www.cidr-report.org/as2.0/", jul 2014.

[37] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460 (Draft Standard), Internet Engineering Task Force, Dec. 1998.

[38] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing Middlebox Interference with Tracebox," in *Proceedings of the 13th ACM Internet Measurement Conference*, 2013.

[39] D. Zhou, B. Fan, H. Lim, M. Kaminsky, and D. G. Andersen, "Scalable, High Performance Ethernet Forwarding with CuckooSwitch," in *Proceedings of the 9th ACM Conference on emerging Network EXperiments and Technologies*, 2013.

[40] S. Han, K. Jang, K. Park, and S. Moon, “PacketShader: A GPU-accelerated Software Router,” in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010.

[41] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, “RouteBricks: Exploiting Parallelism to Scale Software Routers,” in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, 2009.

[42] “Data Plane Development Kit,” ”http://dpdk.org”.

[43] S. Gueron, “Intel Advanced Encryption Standard (AES) New Instruction Set,” ”https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf”, Mar. 2010.

[44] M. Bellare, J. Kilian, and P. Rogaway, “The Security of the Cipher Block Chaining Message Authentication Code,” *J. Comput. Syst. Sci.*, Dec. 2000.

[45] RIPE, “Routing Information Service Raw Data,” ”http://www.ripe.net/data-tools/stats/ris/ris-raw-data”, May 2015.

[46] “Intel Digital Random Number Generator (DRNG) Software Implementation,” ”http://intel.ly/1HOHBOh”, jul 2014.

[47] “Performance Impact of Intel Secure Key on openSSL,” ”http://intel.ly/1EJA2Ib”, jul 2014.

[48] P. Gupta, S. Lin, and N. McKeown, “Routing Lookups in Hardware at Memory Access Speeds,” in *Proceedings of the 17th IEEE Conference on Computer Communications*, Mar. 1998.

[49] K. Argyraki, P. Maniatis, O. Irzak, S. Ashish, and S. Shenker, “Loss and Delay Accountability for the Internet,” in *Proceedings of the 15th IEEE International Conference on Network Protocols*, Oct. 2007.

[50] “Segment Routing,” ”http://www.segment-routing.net/”, 2015.

[51] J. Van der Merwe, A. Cepleanu, K. D’Souza, B. Freeman, A. Greenberg, D. Knight, R. McMillan, D. Moloney, J. Mulligan, H. Nguyen, M. Nguyen, A. Ramarajan, S. Saad, M. Satterlee, T. Spencer, D. Toll, and S. Zelingher, “Dynamic Connectivity Management with an Intelligent Route Service Control Point,” in *Proceedings of the SIGCOMM Workshop on Internet Network Management*, 2006.

[52] I. Cunha, R. Teixeira, and C. Diot, “Measuring and Characterizing End-to-end Route Dynamics in the Presence of Load Balancing,” in *Proceedings of the 12th Conference on Passive and Active Measurement*, 2011.

[53] M. S. Kang, S. B. Lee, and V. D. Gligor, “The Crossfire Attack,” in *Proceedings of the 33rd IEEE Symposium on Security and Privacy*, 2013.

[54] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing,” RFC 2827 (Best Current Practice), Internet Engineering Task Force, May 2000.

[55] F. Baker and P. Savola, “Ingress Filtering for Multihomed Networks,” RFC 3704 (Best Current Practice), Internet Engineering Task Force, Mar. 2004.

[56] S. Zargar, J. Joshi, and D. Tipper, “A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks,” *Communications Surveys Tutorials*, apr 2013.

[57] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service: Attack and Defense Mechanisms (Radia Perlman Computer Networking and Security)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[58] A. Li, X. Liu, and X. Yang, “Bootstrapping Accountability in the Internet We Have,” in *Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation*, 2011.

[59] D. R. Simon, S. Agarwal, and D. A. Maltz, “As-based Accountability as a Cost-effective DDoS Defense,” in *Proceedings of the 1st Workshop on Hot Topics in Understanding Botnets*, 2007.

[60] W. Zhou, Q. Fei, A. Narayan, A. Haeberlen, B. T. Loo, and M. Sherr, “Secure Network Provenance,” in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, 2011.

[61] A. Haeberlen, P. Kouznetsov, and P. Druschel, “PeerReview: Practical Accountability for Distributed Systems,” in *Proceedings of the 21st ACM Symposium on Operating Systems Principles*, 2007.

[62] A. Haeberlen, I. Avramopoulos, J. Rexford, and P. Druschel, “NetReview: Detecting when Interdomain Routing Goes Wrong,” in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, 2009.

[63] D. X. Song and A. Perrig, “Advanced and Authenticated Marking Schemes for IP Traceback,” in *Proceedings of the 20th IEEE Conference on Computer Communications*, 2001.

[64] C. Jin, H. Wang, and K. G. Shin, “Hop-count Filtering: An Effective Defense Against Spoofed DDoS Traffic,” in *Proceedings of the 10th ACM Conference on Computer and Communication Security*, 2003.

[65] M. Natu and J. Mirkovic, “Fine-grained Capabilities for Flooding DDoS Defense Using Client Reputations,” in *Proceedings of the 2007 Workshop on Large Scale Attack Defense*, 2007.

[66] X. Liu, X. Yang, and Y. Xia, “NetFence: Preventing Internet Denial of Service from Inside out,” in *Proceedings of the ACM SIGCOMM Conference on Data Communication*, 2010.